COMMUNICATION PROCESS BY CONNECTING SERVER END IN SERIES WITH SYSTEM UNDER VERIFICATION IN A NETWORK

FIELD OF THE INVENTION

The present invention relates to network communication and more particularly to a communication process by connecting server end in series with system under verification rather than through adapter in a network based environment.

BACKGROUND OF THE INVENTION

Conventionally, it is required to install a network adapter in each of server end and device (e.g., notebook computer) to be processed (hereinafter called system under verification (SUV)) in an assembly line. As such, the device in server end may be capable of performing a variety of tests on the SUV. However, the previous design is disadvantageous for being time consuming and tedious. To the worse, the cost of such design is relatively high in a mass production environment.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a communication process by connecting a server end in series with a system under verification (SUV) in a network. The process initializes a communication port connected to server end and SUV and associated parameters through a computer in server end and SUV. Then creates a required thread and an associated interrupt program in server end and SUV respectively. When data has been received, the received data package is stored in an embedded buffer in server end or SUV until a complete data package is stored in the buffer. Next, data is transmitted through a

predetermined data transmission module. By continuing this process, it is possible to transmit data by connecting server end in series with SUV through the connected communication port rather than network adapter. It is advantageous for being simple and cost effective.

It is another object of the present invention to provide a communication process by connecting a server end in series with a system under verification in a network. When data has been received, the computer in server end or SUV determines whether the received data is complete based on the head of the data package. If not, process aborts. If yes, store the complete data in a receiving buffer. When the receiving of data has been completed a suitable processing is performed on the data package based on the data type thereof.

The above and other objects, features and advantages of the present invention will become apparent from the following detailed description taken with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a communication process by connecting server end in series with system under verification in a network according to the invention;

FIG. 2A is a flow chart diagram illustrating the steps performed in the initialization module of server end of FIG. 1;

FIG. 2B is a flow chart diagram illustrating the steps performed in the initialization module of SUV of FIG. 1;

FIG. 3 is a flow chart diagram illustrating the steps performed in either the thread of FIG. 2A or the interrupt program of FIG. 2B;

FIG. 4 is a flow chart diagram illustrating the process of sending data from server to SUV or from SUV to server; and

FIG. 5 is a flow chart diagram illustrating the process performed by server or SUV when data has been received.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring to FIG. 1, there is shown a block diagram of a communication process by connecting a server end (e.g., server) in series with a system under verification (SUV) (e.g., device to be processed in an assembly line) in a network. Data is communicated between server and SUV. In sending data, first initialize a communication port n which is connected to server or SUV. Next, the computer will pack the data to be sent so as to be in compliance with a predetermined transmission protocol. Then send data to a predetermined buffer in data transmission module in server or SUV. Finally, data is sent. For example, as data sent from server to SUV, data is first sent to initialization module in the SUV. Then data is sent to data receiving module. The received data will be stored in a predetermined buffer prior to sending to cleaning module. Finally, associated head contained in data is deleted in cleaning module so as to obtain the original data sent from server. As in the case of data sent from SUV to server, data is first sent to initialization module in the server through one of a plurality of pairs of communication port n and thread n (where n is 1, 2,...N). Then data is sent to data receiving module. The received data will be stored in a predetermined buffer prior to sending to cleaning module. Finally, the associated head contained in data is deleted in cleaning module so as to obtain the original data sent from server. By continuing this process, it is possible to transmit data between server and SUV through connected communication port rather than through network adapter. It is advantageous for being simple and cost effective. Each of initialization module, data receiving module, and data transmission module is detailed below.

Referring to FIG. 2A, there is shown a flow chart diagram illustrating the steps performed in the initialization module of server of FIG. 1. Computer is commanded to assign a data storage buffer in server to data transmission module and data receiving module respectively. Then it is determined whether the assignment is succeeded. If the assignment fails, process will abort immediately. Otherwise, a communication port is assigned to computer based on an embedded communication port parameter. Further, communication port is initialized immediately. Next, a storage associated with communication port is initialized based on a specified baud rate (i.e., data transfer rate). Then a thread is created. Finally, it is determined whether the initialization and thread creation are succeeded. If it fails, process will abort immediately. Otherwise, process will end normally.

Referring to FIG. 2B, there is shown a flow chart diagram illustrating the steps performed in the initialization module of SUV of FIG. 1. Computer is commanded to assign a data storage buffer in server to data transmission module and data receiving module respectively. Then it is determined whether the assignment is succeeded. If the assignment fails, process will abort immediately. Otherwise, a communication port is assigned to computer based on an embedded communication port parameter. Further, communication port is initialized immediately. Next, a storage associated with communication port is initialized based on a specified baud rate (i.e., data transfer rate). Then an interrupt program is created based on the thread in server. Finally, it is determined whether the initialization and interrupt program creation are succeeded. If it fails, process will abort immediately. Otherwise, process will end normally.

Referring to FIG. 3, there is shown a flow chart diagram illustrating the steps performed in either the thread of FIG. 2A or the interrupt program of FIG. 2B.

Computer in server or SUV continuously monitors the status of communication port connected to server and SUV for determining whether data has been transmitted to the communication port. If no data received, process returns to above monitoring step. If yes, search a complete data package in the receiving buffer. If yes, process returns to above monitoring step. If not, receive data based on the head (including flag, data type, check bits, and etc.) of the data package. Then it is determined whether the data package is complete (i.e., whether a set value is contained therein). If not, process returns to above monitoring step. If yes, retrieve the data size bit in order to know the size of the head and data, and the set value of the data package contained in the head. Then, it is determined whether the value of data is zero. If yes, process returns to above monitoring step. If not, receive data based on data size of the data package. Finally, perform a suitable processing on the data based on the data type thereof.

Referring to FIG. 4, there is shown a flow chart diagram illustrating the process of sending data from server to SUV or from SUV to server. Computer in server or SUV may issue a transmission request to SUV or server (step 401). The computer is waiting a reply therefrom (step 402). Then it is determined whether there is a reply from server or SUV (step 403). If not, it is determined whether the waiting is within an acceptable limit (step 406). If the waiting is still within an acceptable limit, process loops back to step 403. If not, it is determined whether the times of requesting transmission has reached a predetermined value (step 407). If yes, process loops back to step 401 to start again. If not, process will abort immediately. Otherwise if there is a reply from server or SUV, computer will receive the request and transmit the message contained in the request to server or SUV for receiving (step 404). Next, it is determined whether the receiving end accepts the request (step 405). If not, process will abort

immediately. If yes, computer begins to transmit data (step 408). Computer may determine whether there is a reply from the receiving end at the same time (step 409). If not, it is determined whether the waiting is within an acceptable limit (step 411). If the waiting is still within an acceptable limit, process loops back to step 409. If not, it is determined whether the times of requesting receiving has reached a predetermined value (step 412). If yes, process loops back to step 408. If not, process will abort immediately. Otherwise, if there is a reply from the receiving end, computer may determine whether the transmission has ended (step 410). If not, process loops back to step 408. If the transmission has ended, an end of transmission flag is sent to receiving end (step 413). Then it is determined whether there is a reply from receiving end with respect to the end of transmission flag (step 414). If no reply, it is then determined whether the waiting is within an acceptable limit (step 415). If the waiting is still within an acceptable limit, process loops back to step 414. If not, it is determined whether the times of requesting the receiving end to reply has reached a predetermined value (step 416). If yes, process loops back to step 413. If not, process will abort immediately. Otherwise, if there is a reply from the receiving end with respect to the end of transmission flag, process will end normally.

Referring to FIG. 5, there is shown a flow chart diagram illustrating the process performed by server or SUV when data has been received. First, computer in server or SUV may determine whether the received data is in compliance with the data type contained in the data package. If not, process ends immediately. If yes, it is determined whether user buffer has been full. If yes, process ends immediately. If not, write the received data from the receiving buffer into the user buffer. Finally, it is determined whether end of transmission has been received. If yes, process ends immediately. If not, process loops back to beginning.

While the invention has been described by means of specific embodiments, numerous modifications and variations could be made thereto by those skilled in the art without departing from the scope and spirit of the invention set forth in the claims.